Theoretical Computer Science 122 (1994) 119–136
Elsevier

Realizability interpretation of coinductive definitions and program synthesis with streams

Makoto Tatsuta

Research Institute of Electrical Communication, Tohoku University, 2-1-1 Katahira, Sendai 980, Japan

Abstract

Tatsuta, M., Realizability interpretation of coinductive definitions and program synthesis with streams, Theoretical Computer Science 122 (1994) 119–136.

The main aim of the paper is to construct a logical system in which properties of programs can be formalized for verification, synthesis and transformation. The paper has two main points. One point is a realizability interpretation of coinductive definitions of predicates. The other point is extraction of programs which treat streams. An untyped predicative theory TID_ν is presented, which has the facility of coinductive definitions of predicates and is based on constructive logic. Properties defined by the greatest fixed point, such as streams and the extensional equality of streams, can be formalized by the facility of coinductive definitions of predicates in TID_ν. A q-realizability interpretation for TID_ν is defined and the soundness of the interpretation is proved. By the realizability interpretation, a program which treats streams can be extracted from a proof of its specification in TID_ν. A general program extraction theorem and a stream program extraction theorem are presented.

1. Introduction

Our main aim is to construct a logical system in which we can formalize properties of programs for verification, synthesis and transformation. In the paper, we concentrate on formalization of programs with streams and present a theory TID_ν.

Coinductive definitions are very important for this purpose. Properties of streams are represented semantically by the greatest fixed point. The predicate representing what a stream is and the extensional equality of streams are defined semantically by

Correspondence to: M. Tatsuta, Research Institute of Electrical Communication, Tohoku University, 2-1-1 Katahira, Sendai 980, Japan. Email: tatsuta@riec.tohoku.ac.jp.

0304-3975/94/$07.00

the greatest fixed point. The properties defined by the greatest fixed point can be formalized by coinductively defined predicates and coinduction.

μ-calculus has been studied to formalize programs with streams for verification [3]. μ-calculus has the facility of coinductive definitions of predicates and coinduction and is based on classical logic.

In the paper, we present the theory TID_ν, which has the facility of coinductive definitions of predicates and coinduction and is based on constructive logic. By these facilities we can formalize properties of programs with streams in TID_ν.

Our theory TID_ν is based on constructive logic because we want to use the facility of program extraction by realizability for TID_ν. Program extraction is one of the benefits we get when we use a constructive formal theory to formalize properties of programs. Program extraction is to get a program from a constructive proof of its specification formula. One method of program extraction is to use a realizability interpretation. In PX [4], for example, a LISP program is extracted from a proof of its specification formula by a realizability interpretation.

By the facility of coinductive definitions of predicates and a realizability interpretation, we can synthesize programs with streams naturally in TID_ν using theorem proving techniques.

The paper has two main points. One point is a realizability interpretation of coinductive definitions. The other point is extraction of programs with streams.

We present the untyped predicative theory TID_ν, which has coinductive definitions of predicates and is based on constructive logic. We define a $\boldsymbol{q}$-realizability interpretation for TID_ν. We show that the realizability interpretation is sound. We present a general program extraction theorem and a stream program extraction theorem.

The soundness theorem was also proved in [5]. Both works are independent.

In Section 2, we define the theory TID_ν. In Section 3, we briefly explain how useful the facility of coinductive definitions of predicates is to formalize streams. In Section 4, we discuss models of TID_ν and prove its consistency. In Section 5, we present the $\boldsymbol{q}$-realizability interpretation for TID_ν and prove the soundness theorem. In Section 6, we give a general program extraction theorem, a stream program extraction theorem for TID_ν and an example of program synthesis.

2. Theory TID_ν

We present the theory TID_ν in this section. It is the same as Beeson's EON [1] except for the axioms of coinductive definitions of predicates.

In the paper, we choose λ-calculus with pairing and natural numbers as the target programming language for simplicity, since we want to concentrate on the topic of coinductive definitions of predicates. We suppose that the evaluation strategy of terms is lazy or call-by-name because we represent a stream by an infinite list, which is a non-terminating term. We also omit the formalization of the lazy or call-by-name evaluation strategy in TID_ν for simplicity.

Definition 2.1 (*Language of TID$_v$*). The language of TID$_v$ is based on a first-order language and λ-calculus, but is extended for coinductive definitions of predicates.

The constants are

$$\boldsymbol{p}, \boldsymbol{p}_0, \boldsymbol{p}_1, 0, \boldsymbol{s}_N, \boldsymbol{p}_N, \boldsymbol{d}.$$

We choose λ-calculus with pairing and natural numbers as the target programming language for simplicity. We have natural numbers as primitives, which are given by 0, the successor function $\boldsymbol{s}_N$ and the predecessor function $\boldsymbol{p}_N$. We also have pairing functions $\boldsymbol{p}, \boldsymbol{p}_0$ and $\boldsymbol{p}_1$ built-in, which correspond to cons, car and cdr, respectively, in LISP. $\boldsymbol{d}$ is a combinator judging equality of natural numbers and corresponds to an if–then–else statement in a usual programming language.

We have only one function symbol:

App

whose arity is 2. It means a functional application of terms.

Terms t are defined as follows:

$$t ::= x \mid c \mid \mathrm{App}(t, t) \mid \lambda x.t,$$

where x stands for a variable and c stands for a constant.

For terms s, t, we abbreviate $\mathrm{App}(s, t)$ as st. For terms s, t, we also use abbreviations $\langle s, t\rangle \equiv \boldsymbol{p}st$, $t_0 \equiv \boldsymbol{p}_0 t$ and $t_1 \equiv \boldsymbol{p}_1 t$. $\langle x_0, \ldots, x_n\rangle$ denotes $\langle x_0, \langle \ldots \langle x_{n-1}, x_n\rangle \ldots\rangle\rangle$.

The predicate symbols are

$$\bot, N, =.$$

$\bot$ means contradiction. $N(a)$ means that a is a natural number. $a = b$ means that a is equal to b.

We have predicate variables, which a first-order language does not have. The predicate variables are

$$X, Y, Z, \ldots, X^*, Y^*, Z^*, \ldots$$

Each predicate variable has a fixed arity.

We abbreviate $\mathrm{Y}(\lambda x.t)$ as $\mu x.t$, where $\mathrm{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

Definition 2.2 (*Formula*). We define a formula A, a set $S_+(A)$ of predicate variables which occur positively in A and a set $S_-(A)$ of predicate variables which occur negatively in A.

(1) If a, b are terms,

$$\bot, \quad N(a), \quad a = b$$

are formulas. Then

$$S_+(\bot)=S_-(\bot)=\phi,$$

$$S_+(N(a))=S_-(N(a))=\phi,$$

$$S_+(a=b)=S_-(a=b)=\phi.$$

(2) If X is a predicate variable whose arity is n and $t_1, \ldots, t_n$ are terms, $X(t_1, \ldots, t_n)$ is a formula and

$$S_+(X(t_1, \ldots, t_n))=\{X\},$$

$$S_-(X(t_1, \ldots, t_n))=\phi.$$

(3) $A\&B$, $A \vee B$, $A \to B$, $\forall x A$, $\exists x A$ are formulas if A and B are formulas in the same way as a first-order language. Then

$$S_+(A\&B)=S_+(A \vee B)=S_+(A) \cup S_+(B),$$

$$S_-(A\&B)=S_-(A \vee B)=S_-(A) \cup S_-(B),$$

$$S_+(A \to B)=S_-(A) \cup S_+(B),$$

$$S_-(A \to B)=S_+(A) \cup S_-(B),$$

$$S_+(\forall x A)=S_+(\exists x A)=S_+(A),$$

$$S_-(\forall x A)=S_-(\exists x A)=S_-(A).$$

(4) $(\nu X.\lambda x_1 \ldots x_n.A)\,(t_1, \ldots, t_n)$ is a formula, where X is a predicate variable whose arity is n, A is a formula, $t_1, \ldots, t_n$ are terms and X is not in $S_-(A)$. Then

$$S_+((\nu X.\lambda x_1 \ldots x_n.A)\,(t_1, \ldots, t_n))=S_+(A)-\{X\},$$

$$S_-((\nu X.\lambda x_1 \ldots x_n.A)\,(t_1, \ldots, t_n))=S_-(A).$$

The last case corresponds to coinductively defined predicates. Note that X and $x_1, \ldots, x_n$ may occur freely in A. The intuitive meaning of the formula

$$(\nu X.\lambda x_1 \ldots x_n.A(X, x_1, \ldots, x_n))\,(t_1, \ldots, t_n)$$

is as follows: Let P be a predicate of arity n such that P is the greatest solution of the equation

$$P(x_1, \ldots, x_n) \leftrightarrow A(P, x_1, \ldots, x_n).$$

Then $(\nu X.\lambda x_1 \ldots x_n.A(X, x_1, \ldots, x_n))\,(t_1, \ldots, t_n)$ means $P(t_1, \ldots, t_n)$ intuitively.

We abbreviate a sequence as a bold type symbol: for example, $x_1, \ldots, x_n$ as $\boldsymbol{x}$.

Example 2.3. We give an example of a formula. We assume the arity of a predicate variable P is 1. Then

$$(\nu P.\lambda x.x=\langle x_0, x_1\rangle \;\&\; x_0=0 \;\&\; P(x_1))\,(x)$$

is a formula.

Among many axioms and inference rules of TID_ν, we discuss only axioms of coinductive definitions of predicates. The rest of the axioms and inference rules are almost the same as EON [1] and we only list them in Appendix A.

Definition 2.4 (*Coinductive definitions*). Let $\nu \equiv \nu P.\lambda \boldsymbol{x}.A(P)$ where $\boldsymbol{x}$ is a sequence of variables whose length is the same as the arity of a predicate variable P and $A(P)$ is a formula displaying all the free occurrences of P in a formula A. Suppose that $C(\boldsymbol{x})$ is a formula displaying all the free occurrences of variables $\boldsymbol{x}$ in the formula.

We have the following axioms:

$$\forall \boldsymbol{x}(\nu(\boldsymbol{x}) \to A(\nu)), \tag{$\nu 1$}$$

$$\forall \boldsymbol{x}(C(\boldsymbol{x}) \to A(C)) \to \forall \boldsymbol{x}(C(\boldsymbol{x}) \to \nu(\boldsymbol{x})). \tag{$\nu 2$}$$

$\nu P.\lambda x.A(P)$ means the greatest fixed point of the function from a predicate P to a predicate $\lambda x.A(P)$.

We define a theory TID^- as the theory TID_ν except that the theory TID^- does not have the two axioms ($\nu 1$) and ($\nu 2$) of coinductive definitions of predicates.

3. Coinductive definitions of predicates

We explain coinductive definitions of TID_ν and show some examples of formalization of streams by coinductive definitions.

Proposition 3.1. *Let* ν *be* $\nu X.\lambda \boldsymbol{x}.A(X)$. *Then*

$$\forall \boldsymbol{x}(\nu(\boldsymbol{x}) \leftrightarrow A(\nu)) \tag{$\nu 1'$}$$

holds.

Proof. By ($\nu 1$), we get $\nu(\boldsymbol{x}) \to A(\nu)$. By letting C be $\lambda \boldsymbol{x}.A(\nu)$ in ($\nu 2$), $A(\nu) \to \nu(\boldsymbol{x})$ holds. □

This proposition shows that $\nu P.\lambda \boldsymbol{x}.A(P)$ is the solution of the following recursive equation of a predicate P:

$$P(\boldsymbol{x}) \leftrightarrow A(P).$$

The axiom ($\nu 2$) says that $\nu P.\lambda \boldsymbol{x}.A(P)$ is the greatest solution of this equation or the greatest fixed point of the function $\lambda P.\lambda \boldsymbol{x}.A(P)$.

Streams can be formalized by coinductive definitions [3]. Therefore we can formalize streams in TID_ν.

We represent a stream by an infinite list $\langle a, s \rangle$ constructed by pairing where a is the first element of the stream, and s is the rest of the stream. In this representation, if s is a stream, we can get the first element of s by s_0 and the rest by s_1.

We present an example of bit streams. A bit stream is a stream whose elements are 0 or 1. We will define a predicate $BS(x)$ which means that x is a bit stream. When we write down a formula $BS(x)$ in a naive way, BS itself occurs in the body of the definition as follows:

$$BS(x) \leftrightarrow x = \langle x_0, x_1 \rangle \ \& \ (x_0 = 0 \vee x_0 = 1) \ \& \ BS(x_1).$$

BS is a solution P of the following equation for a predicate P

$$P(x) \leftrightarrow x = \langle x_0, x_1 \rangle \ \& \ (x_0 = 0 \vee x_0 = 1) \ \& \ P(x_1) \tag{1}$$

or a fixed point of the function

$$\lambda P.\lambda x.x = \langle x_0, x_1 \rangle \ \& \ (x_0 = 0 \vee x_0 = 1) \ \& \ P(x_1). \tag{2}$$

There may be many solutions P for (1). For example, $\lambda x.\bot$ is one solution of (1), although it is not our intended solution. $\lambda x.\bot$ is the least solution. Our intended solution is the greatest solution of (1) or the greatest fixed point of (2). Hence we have the solution in TID_ν and it is represented as follows:

$$BS \equiv \nu P.\lambda x.x = \langle x_0, x_1 \rangle \ \& \ (x_0 = 0 \vee x_0 = 1) \ \& \ P(x_1).$$

Let $\bar{0}$ be $\mu s.\langle 0, s \rangle$. $\bar{0}$ represents the zero stream whose elements are all 0. We can show $BS(\bar{0})$ by the coinduction axiom $(\nu 2)$. Let C be $\lambda x.(x = \bar{0})$ in $(\nu 2)$, then we have

$$\forall x(x = \bar{0} \to x = \langle x_0, x_1 \rangle \ \& \ (x_0 = 0 \vee x_0 = 1) \ \& \ x_1 = \bar{0}) \to \forall x(x = \bar{0} \to BS(x)).$$

By definition of $\bar{0}$,

$$\forall x(x = \bar{0} \to x = \langle x_0, x_1 \rangle \ \& \ (x_0 = 0 \vee x_0 = 1) \ \& \ x_1 = \bar{0})$$

holds and we have

$$\forall x(x = \bar{0} \to BS(x)).$$

Let $x = \bar{0}$ then we get $BS(\bar{0})$.

The coinductive definitions of predicates also play an important role in representing predicates which describe properties of streams [3, 6]. We will define the extensional equality $s \approx t$ for streams s and t. This equality can be represented by the coinductive definitions of predicates. The relation $\approx$ is the greatest solution of the following equation for a predicate P:

$$P(x, y) \leftrightarrow x_0 = y_0 \ \& \ P(x_1, y_1)$$

Therefore the relation $\approx$ can be formalized in TID_ν as follows:

$$\approx \equiv \nu P.\lambda xy.x_0 = y_0 \ \& \ P(x_1, y_1).$$

4. Models of TID_ν

We will briefly explain the semantics of TID_ν by giving its intended model.

We will use classical set theory and the well-known greatest fixed point theorem for model construction in this section.

Theorem 4.1 (Greatest fixed point). *Suppose S be a set and $p(S)$ be the power set of S. If $f: p(S) \to p(S)$ is a monotone function, there exists a such that $a \in p(S)$ and*

(1) $f(a) = a$,

(2) *For any $b \in p(S)$, if $b \subset f(b)$, then $b \subset a$.*

a is abbreviated as $\mathrm{gfp}(f)$.

We will construct a model M' of TID_ν by extending an arbitrary model M of TID^-. Our intended model of TID^- is the closed total term model whose universe is the set of closed terms [1]. We denote the universe by U.

We will define $\rho \models A$ in almost the same way as for first-order logic, where A is a formula and ρ is an environment which assigns an element of U to a first-order variable and a subset of U^n to a predicate variable of arity n and which covers all the free first-order variables and all the free predicate variables of A. We only present the definition for the case $(\nu P.\lambda \boldsymbol{x}.A(P))(t)$.

Define F as follows:

$$|\boldsymbol{x}| = n,$$

$$F: p(U^n) \to p(U^n),$$

$$F(X) = \{\boldsymbol{x} \in U^n \mid \rho[P := X] \models A(P)\},$$

where $\rho[P := X]$ is defined as follows:

$$\rho[P := X](P) = X,$$

$$\rho[P := X](x) = \rho(x) \quad \text{if } x \text{ is not } P.$$

Then $\rho \models (\nu P.\lambda \boldsymbol{x}.A(P))(\boldsymbol{t})$ is defined as $\boldsymbol{t} \in \mathrm{gfp}(F)$. Note that F is monotone since a predicate variable P occurs only positively in $A(P)$.

Theorem 4.2. *If $\mathrm{TID}_\nu \vdash A$, then $\rho \models A$ for any environment ρ which covers all the free variables of A.*

Theorem 4.3. TID_ν *is consistent.*

5. q-realizability interpretation for TID_ν

We will explain the motivation of our realizability. We start with the usual q-realizability and try to interpret $(\nu P.\lambda x.A(P))(x)$. Let ν be $\nu P.\lambda x.A(P)$. Then

$v(x) \leftrightarrow A(v, x)$ holds. We want to treat $v(x)$ and $A(v, x)$ in the same manner. So we require $(e\ \boldsymbol{q}\ v(x)) \leftrightarrow (e\ \boldsymbol{q}\ A(v, x))$. Therefore it is natural to define $(e\ \boldsymbol{q}\ v(x))$ as $v^*(e, x)$, where $v^*(e, x)$ is the greatest solution of the recursive equation for a predicate variable X^*:

$$X^*(e, x) \leftrightarrow (e\ \boldsymbol{q}\ A(v, x))\ [(r\ \boldsymbol{q}\ v(y)) := X^*(r, y)],$$

where $[(r\ \boldsymbol{q}\ v(y)) := X^*(r, y)]$ of the right-hand side means replacing each subformula $(r\ \boldsymbol{q}\ v(y))$ by a subformula $X^*(r, y)$ in the formula $(e\ \boldsymbol{q}\ A(v, x))$. We get the following definition of our realizability by describing this idea syntactically.

Our realizability in this paper is an extension of Grayson's realizability. We can also define the usual Kleene-style $\boldsymbol{q}$-realizability of coinductively defined predicates in the same way as in the paper.

Definition 5.1 (*Harrop formula*). (1) Atomic formulas $\bot$, $N(a)$ and $a = b$ are Harrop.

(2) If A and B are Harrop, then $A\ \&\ B$, $C \to B$, $\forall x A$ and $(\nu P.\lambda \boldsymbol{x}.A)(\boldsymbol{t})$ are also Harrop.

Since a Harrop formula does not have computational meanings, we can simplify the $\boldsymbol{q}$-realizability interpretation of them.

Definition 5.2 (*Abstract*). (1) A predicate symbol of arity n is an abstract of arity n.

(2) A predicate variable of arity n is an abstract of arity n.

(3) If A is a formula, $\lambda x_1 \ldots x_n.A$ is an abstract of arity n.

(4) If $(\nu X.\lambda x_1 \ldots x_n.A)(x_1, \ldots, x_n)$ is a formula, $\nu X.\lambda x_1 \ldots x_n.A$ is an abstract of arity n.

We identify $(\lambda x_1 \ldots x_n.A)(t_1, \ldots, t_n)$ with $A[x_1 := t_1, \ldots, x_n := t_n]$, where $[x_1 := t_1, \ldots, x_n := t_n]$ denotes a simultaneous substitution.

Definition 5.3. (*q-realizability interpretation*). Suppose A is a formula, $P_1, \ldots, P_n$ is a sequence of predicate variables whose arities are $m_1, \ldots, m_n$, respectively, and $F_1, G_1, \ldots, F_n, G_n$ is a sequence of abstracts whose arities are $m_1, m_1 + 1, \ldots, m_n, m_n + 1$, respectively.

$$(e\ \boldsymbol{q}_{P_1, \ldots, P_n}[F_1, G_1, \ldots, F_n, G_n]\ A)$$

is defined by induction on the construction of A as follows.

We abbreviate $\boldsymbol{q}_{P_1, \ldots, P_n}[F_1, G_1, \ldots, F_n, G_n]$ as $\boldsymbol{q}'$, $\boldsymbol{q}_{P_1, \ldots, P_n, P}[F_1, G_1, \ldots, F_n, G_n, F, G]$ as $\boldsymbol{q}'_P[F, G]$, $F_1, \ldots, F_n$ as $\boldsymbol{F}$ and $P_1, \ldots, P_n$ as $\boldsymbol{P}$.

(1) $(e\ \boldsymbol{q}'\ A) \equiv e = 0\ \&\ A_P[\boldsymbol{F}]$, where A is Harrop.

(2) $(e\ \boldsymbol{q}'\ P_i(\boldsymbol{t})) \equiv F_i(\boldsymbol{t})\ \&\ G_i(e, \boldsymbol{t})$.

(3) $(e\ \boldsymbol{q}'\ Q(\boldsymbol{t})) \equiv Q(\boldsymbol{t})\ \&\ Q^*(e, \boldsymbol{t})$, where $Q \not\equiv P_i$ $(1 \leqslant i \leqslant n)$.

(4) $(e\ \boldsymbol{q}'\ A\ \&\ B) \equiv (e_0\ \boldsymbol{q}'\ A)\ \&\ (e_1\ \boldsymbol{q}'\ B)$.

(5) $(e\ \boldsymbol{q}'\ A \vee B) \equiv N(e_0)\ \&\ (e_0 = 0 \to (e_1\ \boldsymbol{q}'\ A))\ \&\ (e_0 \neq 0 \to (e_1\ \boldsymbol{q}'\ B))$.

(6) $(e\ \boldsymbol{q}'\ A \to B) \equiv (A \to B)_P\ [\boldsymbol{F}]\ \&\ \forall q((q\ \boldsymbol{q}'\ A) \to (eq\ \boldsymbol{q}'\ B))$.

(7) $(e\ \boldsymbol{q}'\ \forall x A(x)) \equiv \forall x (ex\ \boldsymbol{q}'\ A(x))$.
(8) $(e\ \boldsymbol{q}'\ \exists x A(x)) \equiv (e_1\ \boldsymbol{q}'\ A(e_0))$.
(9) $(e\ \boldsymbol{q}'\ (\nu X.\lambda \boldsymbol{x}.A(X))(\boldsymbol{t})) \equiv (\nu X^*.\lambda r \boldsymbol{x}.(r\ \boldsymbol{q}'_X[\nu_P[\boldsymbol{F}], X^*]\ A(X)))(e, \boldsymbol{t})$, where $\nu \equiv \nu X.\lambda \boldsymbol{x}.A(X)$.

In the above definition, ${}_{P_1,\ldots,P_n}[F_1, G_1, \ldots, F_n, G_n]$ means a substitution. Our realizability interpretation is something like a realizability interpretation with substitution.

Proposition 5.4. *Let* $\nu \equiv \nu P.\lambda \boldsymbol{x}.A(P)$.
(1) $\forall x r((r\ \boldsymbol{q}\ \nu(\boldsymbol{x})) \leftrightarrow (r\ \boldsymbol{q}\ A(\nu)))$.
(2) $(\lambda x r.r\ \boldsymbol{q}\ \forall \boldsymbol{x}(\nu(\boldsymbol{x}) \to A(\nu)))$ *holds if* $A(X)$ *is not Harrop.* $(0\ \boldsymbol{q}\ \forall \boldsymbol{x}(\nu(\boldsymbol{x}) \to A(\nu)))$ *holds if* $A(X)$ *is Harrop.*

Proof. By the definition of $\boldsymbol{q}$-realizability and ($\nu 1'$). □

Definition 5.5. For a formula A, a predicate variable P and a term f, we define a term $\sigma_A^{P,f}$ by induction on the construction of A as follows:
(1) A is a Harrop formula, then $\sigma_A^{P,f} \equiv \lambda r.r$.
(2) $A \equiv P(\boldsymbol{t})$, then $\sigma_A^{P,f} \equiv \lambda r.f\boldsymbol{t}r$.
(3) $A \equiv Q(\boldsymbol{t})$, then $\sigma_A^{P,f} \equiv \lambda r.r$ if $Q \not\equiv P$.
(4) $A \equiv A_1\ \&\ A_2$, then $\sigma_A^{P,f} \equiv \lambda r.\langle \sigma_{A_1}^{P,f} r_0, \sigma_{A_2}^{P,f} r_1 \rangle$.
(5) $A \equiv A_1 \vee A_2$, then $\sigma_A^{P,f} \equiv \lambda r.\langle r_0, \boldsymbol{d}r_0 0 \sigma_{A_1}^{P,f} \sigma_{A_2}^{P,f} r_1 \rangle$.
(6) $A \equiv A_1 \to A_2$, then $\sigma_A^{P,f} \equiv \lambda r q.\sigma_{A_2}^{P,f}(r(\sigma_{A_1}^{P,f} q))$.
(7) $A \equiv \forall x A_1(x)$, then $\sigma_A^{P,f} \equiv \lambda r x.\sigma_{A_1(x)}^{P,f}(rx)$.
(8) $A \equiv \exists x A_1(x)$, then $\sigma_A^{P,f} \equiv \lambda r.\langle r_0, \sigma_{A_1(r_0)}^{P,f} r_1 \rangle$.
(9) $A \equiv (\nu Q.\lambda \boldsymbol{x}.A_1)(\boldsymbol{t})$, then $\sigma_A^{P,f} \equiv (\mu g.\lambda \boldsymbol{x} r.\sigma_{A_1}^{Q,g}(\sigma_{A_1}^{P,f} r))\boldsymbol{t}$, where $Q \not\equiv P$.

Proposition 5.6. *Let* $\nu \equiv \nu P.\lambda \boldsymbol{x}.A(P)$. If $\nu(\boldsymbol{x})$ *is not Harrop,*

$$\lambda q.\mu f.\lambda \boldsymbol{x} r.\sigma_{A(P)}^{P,f}(q\boldsymbol{x}r)\ \boldsymbol{q}\ \forall \boldsymbol{x}(C(\boldsymbol{x}) \to A(C)) \to \forall \boldsymbol{x}(C(\boldsymbol{x}) \to \nu(\boldsymbol{x}))$$

holds. If $\nu(\boldsymbol{x})$ *is Harrop,*

$$0\ \boldsymbol{q}\ \forall \boldsymbol{x}(C(\boldsymbol{x}) \to A(C)) \to \forall \boldsymbol{x}(C(\boldsymbol{x}) \to \nu(\boldsymbol{x}))$$

holds.

We prove this in Appendix B.

Theorem 5.7 (Soundness theorem). *If* $\mathrm{TID}_\nu \vdash A$, *we can get a term* e *from the proof of* $\vdash A$ *and* $\mathrm{TID}_\nu \vdash (e\ \boldsymbol{q}\ A)$ *holds where all the free variables of* e *are included in all the free variables of* A.

Proof. By induction on the proof of $\vdash A$. The case of the axiom ($\nu 1$) is proved by Proposition 5.4. The case of the axiom ($\nu 2$) is proved by Proposition 5.6. □

6. Program synthesis with streams

In this section, we give a general program extraction theorem, a stream program extraction theorem for TID_ν and an example of program synthesis.

Program synthesis by theorem-proving techniques has been studied in both typed theories [2] and untyped theories [4]. For untyped theories, realizability interpretations are used as the foundations of program synthesis by theorem-proving techniques. In Section 3, we showed that streams and programs which treat streams can be formalized in TID_ν by the facility of coinductive definitions of predicates. In Section 5, we showed that the realizability interpretation can be defined for TID_ν and the interpretation is sound. Hence we can synthesize programs which treat streams by theorem-proving techniques in TID_ν using the realizability interpretation.

We represent streams by infinite lists constructed by pairing. We represent a specification of a program by a formula:

$$\forall x(A(x) \to \exists y B(x, y)),$$

where x is an input, y is an output, $A(x)$ is an input condition and $B(x, y)$ is an input–output relation.

Theorem 6.1 (Program extraction). *Suppose that we prove a specification formula* $\forall x(A(x) \to \exists y B(x, y))$ *of a program in* TID_ν *and we have a realizer* j *such that*

$$\forall x(A(x) \to (jx \boldsymbol{q}\, A(x))).$$

Then we can get a program f *and a proof of*

$$\forall x(A(x) \to B(x, fx))$$

effectively from the proof of the specification formula.

Proof. Since the specification formula is proved in TID_ν, by the soundness theorem (Theorem 5.7) of the $\boldsymbol{q}$-realizability interpretation we have a realizer e such that

$$e \boldsymbol{q}\, \forall x(A(x) \to \exists y B(x, y))$$

holds. Let f be $\lambda x.(ex(jx))_0$. Then the claim holds. □

We can synthesize a program in the following steps:

(1) We write down a specification formula.
(2) We prove the specification formula in TID_ν.
(3) We extract a program from the proof.

The program extraction theorem says that the third step can be automated completely.

Example 6.2. We show an example of the program which gets a stream of natural numbers and returns a stream whose every element is the element of the input stream plus one.

The predicate $NS(x)$ which says that x is a stream of natural numbers can be represented in TID_ν by the facility of coinductive definitions of predicates as follows:

$$NS \equiv \nu X.\lambda x.x = \langle x_0, x_1 \rangle \ \& \ N(x_0) \ \& \ X(x_1).$$

The input condition of the specification is the formula $NS(x)$.

The input–output relation of the specification is the formula $ADD1(x, y)$, which is defined as follows:

$$ADD1 \equiv \nu X.\lambda x y.y_0 = x_0 + 1 \ \& \ X(x_1, y_1).$$

The specification formula is

$$\forall x(NS(x) \to \exists y ADD1(x, y)).$$

We have one problem for this program synthesis method. The coinduction cannot be applied to the part $\forall x(NS(x) \to \cdots)$ in the above example. We cannot prove $\exists y ADD1(x, y)$ by the coinduction in general. Therefore the realizer of the coinduction cannot give a loop structure to the program. On the other hand, the realizer of the induction principle plays an important role in this approach of program synthesis since the realizer corresponds to a loop structure of a program [4, 7]. Therefore we need a new method by which the realizer of the coinduction also corresponds to a loop structure and is useful.

Then we need a more specialized program extraction method for programs with streams by which the coinduction is useful. We give one solution for this problem by the next theorem.

Let $nthcdr \equiv \mu f.\lambda n x.\boldsymbol{dn}0x(f(\boldsymbol{p}_N n)(\boldsymbol{p}_1 x))$. *nthcdr* is the same function as in LISP.

We put two restrictions on the theorem: Let p be a term such that $px = nthcdr\, nx$ for some n depending on x_0. One is that the input condition $A(x)$ must be of the form $(\nu X.\lambda x.x = \langle x_0, x_1 \rangle \ \& \ \tilde{A}(x_0) \ \& \ X(px))(x)$ for some $\tilde{A}$. The other is that the input–output relation $B(x, y)$ must be of the form $(\nu X.\lambda x y.\tilde{B}(x, y_0) \ \& \ X(px, y_1))(x, y)$ for some $\tilde{B}$. These restrictions require that the input condition and the input–output relation are uniform over data, and they are natural when we suppose that the input x and the output y are both streams.

Theorem 6.3 (Stream program extraction). *Let* $nthcdr \equiv \mu f.\lambda n x.\boldsymbol{dn}0x(f(\boldsymbol{p}_N n)x_1)$. *Suppose that the specification formula is* $\forall x(A(x) \to \exists y B(x, y))$,

$$A \equiv \nu X.\lambda x.x = \langle x_0, x_1 \rangle \ \& \ \tilde{A}(x_0) \ \& \ X(px),$$

$$B \equiv \nu X.\lambda x y.\tilde{B}(x, y_0) \ \& \ X(px, y_1),$$

$$p \equiv \lambda x.nthcdr(nx_0)x$$

for some formulas $\tilde{A}(x)$ *and* $\tilde{B}(x, y)$ *and some term* n *such that*

$$\forall x(\tilde{A}(x) \to N(nx))$$

holds and we have a term j such that $\forall x(A(x) \to (jx \; \boldsymbol{q} \; A(x)))$. *Then we define*

$$B^\circ \equiv \nu X.\lambda x.\exists z \tilde{B}(x, z) \,\&\, X(px).$$

If we have e such that

$$e \; \boldsymbol{q} \; \forall x(A(x) \to B^\circ(x)),$$

we can get a term F such that

$$\forall x(A(x) \to B(x, Fx)),$$

where

$$filter \equiv \mu f.\lambda x.\langle x_{00}, fx_1 \rangle,$$

$$F \equiv \lambda x.filter(ex(jx)).$$

We prove this in Appendix C.

By this theorem, we can synthesize a program in the following steps:

(1) We write down a specification formula $\forall x(A(x) \to \exists y B(x, y))$.

(2) We prove the corresponding formula $\forall x(A(x) \to B^\circ(x))$ in TID_ν.

(3) We extract a program $\lambda x.filter(ex(jx))$ from the proof, where e is a realizer of the corresponding formula $\forall x(A(x) \to B^\circ(x))$.

In the second step, we can apply the coinduction to prove the part $B^\circ(x)$ since $B^\circ(x)$ is defined by coinductive definitions. Therefore the realizer of the coinduction can give a loop structure to the program.

Example 6.4. We treat the same example as Example 6.2 again. The specification formula is the formula $\forall x(NS(x) \to \exists y ADD1(x, y))$. Let a term n be $\lambda x.1$ in Theorem 6.3. Then $px = x_1$. Hence the formula $ADD1^\circ(x)$ is

$$ADD1^\circ \equiv \nu X.\lambda x.\exists z(z = x_0 + 1) \,\&\, X(x_1). \tag{3}$$

Therefore the corresponding formula we must prove is

$$\forall x(NS(x) \to ADD1^\circ(x)). \tag{4}$$

If we prove this formula in TID_ν, we can get a program which satisfies the specification by the stream program extraction theorem.

The conditions of the theorem hold for this case. We can put $j \equiv \lambda x.\mu s.\langle 0, s \rangle$ since

$$\forall x(NS(x) \to (\mu s.\langle 0, s \rangle \; \boldsymbol{q} \; NS(x))).$$

We prove (4) in the following way here: Firstly, we prove

$$\forall x(NS(x) \to \exists z(z = x_0 + 1) \,\&\, NS(x_1)). \tag{5}$$

This is proved by letting z be $x_0 + 1$. Secondly, by letting C be NS in ($\nu 2$) for $ADD1^\circ$, we have

$$\forall x(NS(x) \to \exists z(z = x_0 + 1) \,\&\, NS(x_1)) \to \forall x(NS(x) \to ADD1^\circ(x)). \tag{6}$$

Finally, by (5) and (6), we get (4).

We calculate realizers corresponding to the above proofs as follows: The realizer e_1 corresponding to the proof of (5) is

$$e_1 \equiv \lambda xr.\langle\langle x_0+1, 0\rangle, r_{11}\rangle,$$

$$e_1 \; \boldsymbol{q} \; \forall x(NS(x) \to \exists z(z = x_0+1) \;\&\; NS(x_1)).$$

The realizer e_2 corresponding to the proof of (6) is

$$e_2 \equiv \lambda q.\mu f.\lambda xr.\sigma(qxr),$$

$$e_2 \; \boldsymbol{q} \; \forall x(NS(x) \to \exists z(z = x_0+1) \;\&\; NS(x_1)) \to \forall x(NS(x) \to ADD1^\circ(x)),$$

where

$$\sigma \equiv \lambda r.\langle\langle r_{00}, r_{01}\rangle, f x_1 r_1\rangle.$$

The realizer e corresponding to the proof of (4) is

$$e \equiv e_2 e_1,$$

$$e \; \boldsymbol{q} \; \forall x(NS(x) \to ADD1^\circ(x)).$$

We get

$$e = \mu f.\lambda xr.\langle\langle x_0+1, 0\rangle, f x_1 r_{11}\rangle.$$

The extracted program F is

$$Fx = \mathit{filter}(ex(jx)) = \mathit{filter}(fx(\mu s.\langle 0, s\rangle)),$$

where $f \equiv \mu f.\lambda xr.\langle\langle x_0+1, 0\rangle, f x_1 r_{11}\rangle$. Then we have $Fx = (\mu g.\lambda x.\langle x_0+1, g x_1\rangle x$.

This is the program we expect.

Note that the realizer e_2 of the coinduction (6) gives the loop structure to the program F.

Acknowledgment

I am deeply grateful to Professor Masahiko Sato for invaluable discussions and comments. I would also like to thank Mr. Satoshi Kobayashi and Mr. Yukiyoshi Kameyama for careful comments.

Appendix A: Axioms and inference rules of TID$_\nu$

The logical axioms and inference rules are the same as those of usual intuitionistic logic.

Axioms for equality:

$$\forall x(x = x), \tag{E1}$$

$$\forall x, y(x = y \;\&\; A(x) \to A(y)). \tag{E2}$$

Axiom for β-conversion:

$$(\lambda x.a)b = a[x := b], \tag{B1}$$

where $a[x := b]$ means a term obtained from a by replacing every free occurrence of x by b.

Axioms for pairing:

$$\forall x, y(\boldsymbol{p}_0(\boldsymbol{p}xy) = x), \tag{P1}$$

$$\forall x, y(\boldsymbol{p}_1(\boldsymbol{p}xy) = y). \tag{P2}$$

Axioms for natural numbers:

$$N(0), \tag{N1}$$

$$\forall x(N(x) \to N(\boldsymbol{s}_N x)), \tag{N2}$$

$$\forall x(N(x) \to \boldsymbol{p}_N(\boldsymbol{s}_N x) = x), \tag{N3}$$

$$\forall x(N(x) \to \boldsymbol{s}_N x \neq 0), \tag{N4}$$

$$A(0) \ \& \ \forall x(N(x) \ \& \ A(x) \to A(\boldsymbol{s}_N x)) \to \forall x(N(x) \to A(x)). \tag{N5}$$

Axioms for $\boldsymbol{d}$:

$$\forall x, y, a, b(N(x) \ \& \ N(y) \ \& \ x = y \to \boldsymbol{d}xyab = a), \tag{D1}$$

$$\forall x, y, a, b(N(x) \ \& \ N(y) \ \& \ x \neq y \to \boldsymbol{d}xyab = b). \tag{D2}$$

Appendix B: Proof of soundness theorem

Lemma B.1. *Let $\tilde{G}^f$ be $\lambda y\boldsymbol{x}.\exists r(y = f\boldsymbol{x}r \ \& \ G(r, \boldsymbol{x}))$ for an abstract G and a term f and $\boldsymbol{q}'$ be $\boldsymbol{q}_{P_1, \ldots, P_n}[F_1, G_1, \ldots, F_n, G_n]$.*

(1) *If a predicate variable P occurs only positively in a formula A,*

$$(r\, \boldsymbol{q}'_P[F, G]\, A) \to (\sigma_A^{P,f}\, r\, \boldsymbol{q}'_P[F, \tilde{G}^f]\, A).$$

(2) *If a predicate variable P occurs only negatively in a formula A,*

$$(r\, \boldsymbol{q}'_P[F, \tilde{G}^f]\, A) \to (\sigma_A^{P,f}\, r\, \boldsymbol{q}'_P[F, G]\, A).$$

Proof. We suppose that the arity of P is 1 in the proof for simplicity.

We will prove (1) and (2) simultaneously by induction on the construction of A. We will show only some cases, for other cases are proved similarly.

Case A Harrop:

(1) The claim holds since it is $r = 0 \ \& \ A \to (\lambda r.r)r = 0 \ \& \ A$.

(2) Similar to (1).

Case $A \equiv P(t)$:

(1) The claim is $F(t)$ & $G(r,t) \to F(t)$ & $\tilde{G}^f((\lambda r.ftr)r,t)$. Since we have $\tilde{G}^f((\lambda r.ftr)r,t) \leftrightarrow \exists s(ftr = fts$ & $G(s,t))$, the claim holds by putting $s := r$.

(2) This case does not exist because P occurs positively in $P(t)$.

Case $A \equiv v(t)$, *where* $v \equiv vQ.\lambda x.B(P,Q)$:

Let

$$v^* \equiv vQ^*.\lambda rx.(r\ \boldsymbol{q}'_{P,Q}[F, G, v_P[F], Q^*]\ B(P,Q)),$$

$$v_1^* \equiv vQ^*.\lambda rx.(r\ \boldsymbol{q}'_{P,Q}[F, \tilde{G}^f, v_P[F], Q^*]\ B(P,Q)),$$

$$g \equiv \mu g.\lambda xr.\sigma^{Q,g}_{B(P,Q)}(\sigma^{P,f}_{B(P,Q)}\, r).$$

We will show $v^*(r,x) \to v_1^*(\sigma^{P,f}_{v(x)} r, x)$,

By using only rules of NJ, it is equivalent to

$$\forall rx(v^*(r,x) \to v_1^*(\sigma^{P,f}_{v(x)} r, x)),$$

$$\forall rxy(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x) \to v_1^*(y,x))$$

and

$$\forall xy(\exists r(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x)) \to v_1^*(y,x)).$$

By ($v2$), it is sufficient to show

$$\forall xy(\exists r(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x)) \to (y\ \boldsymbol{q}'_{P,Q}[F, \tilde{G}^f, v_P[F],$$
$$\lambda yx.\exists r(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x))]\ B(P,Q)).$$

By using only rules of NJ, it is equivalent to

$$\forall xyr(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x) \to (y\ \boldsymbol{q}'_{P,Q}[F, \tilde{G}^f, v_P[F],$$
$$\lambda yx.\exists r(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x))]\ B(P,Q))$$

and

$$\forall xr(v^*(r,x) \to (\sigma^{P,f}_{v(x)} r\ \boldsymbol{q}'_{P,Q}[F, \tilde{G}^f, v_P[F],$$
$$\lambda yx.\exists r(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x))]\ B(P,Q)). \tag{7}$$

We will now show this. Fix x and r. Assume $v^*(r,x)$. By ($v2$) and $v^*(r,x)$, we have

$$r\ \boldsymbol{q}'_{P,Q}[F, G, v_P[F], v^*]\ B(P,Q).$$

By the induction hypothesis for $B(P,Q)$, we get

$$\sigma^{P,f}_{B(P,Q)}\, r\ \boldsymbol{q}'_{P,Q}[F, \tilde{G}^f, v_P[F], v^*]\ B(P,Q).$$

Using the induction hypothesis for $B(P,Q)$ again, we have

$$\sigma^{Q,g}_{B(P,Q)}(\sigma^{P,f}_{B(P,Q)}\, r)\, \boldsymbol{q}'_{P,Q}[F, \tilde{G}^f, v_P[F], \widetilde{v^{**}}]\ B(P,Q).$$

By $\sigma^{P,f}_{v(x)} \equiv gx$, $\widetilde{v^{*g}} \equiv \lambda yx.\exists r(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x))$ holds. By $\sigma^{Q,g}_{B(P,Q)}(\sigma^{P,f}_{B(P,Q)} r) = \sigma^{P,f}_{v(x)} r$,

$$\sigma^{P,f}_{v(x)} r \ \boldsymbol{q}'_{P,Q}[F, \tilde{G}^f, v_P[F], \lambda yx.\exists r(y = \sigma^{P,f}_{v(x)} r \ \&\ v^*(r,x))] \ B(P,Q)$$

holds and we get the claim (7).

(2) Similar to (1). □

Proof of Proposition 5.6. Suppose that $v(\boldsymbol{x})$ is not Harrop. Let $v \equiv vP.\lambda \boldsymbol{x}.A(P)$.

Assume

$$q \ \boldsymbol{q} \ \forall \boldsymbol{x}(C(\boldsymbol{x}) \to A(C))$$

and let

$$f \equiv \mu f.\lambda \boldsymbol{x} r.\sigma^{P,f}_{A(P)}(q\boldsymbol{x}r).$$

We will show

$$f \ \boldsymbol{q} \ \forall \boldsymbol{x}(C(\boldsymbol{x}) \to v(\boldsymbol{x})).$$

Let $v^*(r, \boldsymbol{x}) \equiv (r \ \boldsymbol{q} \ v(\boldsymbol{x}))$. It is sufficient to show

$$\forall \boldsymbol{x} r((r \ \boldsymbol{q} \ C(\boldsymbol{x})) \to v^*(f\boldsymbol{x}r, \boldsymbol{x})).$$

This is equivalent to

$$\forall \boldsymbol{x} y(\exists r((r \ \boldsymbol{q} \ C(\boldsymbol{x})) \ \&\ y = f\boldsymbol{x}r) \to v^*(y, \boldsymbol{x})).$$

By $(v2)$, it is sufficient to show

$$\forall \boldsymbol{x} y(\exists r((r \ \boldsymbol{q} \ C(\boldsymbol{x})) \ \&\ y = f\boldsymbol{x}r) \to (y \ \boldsymbol{q}_P[v, \lambda yx.\exists r((r \ \boldsymbol{q} \ C(\boldsymbol{x})) \ \&\ y = f\boldsymbol{x}r)] \ A(P))).$$

This is equivalent to

$$\forall \boldsymbol{x} r((r \ \boldsymbol{q} \ C(\boldsymbol{x})) \to (f\boldsymbol{x}r \ \boldsymbol{q}_P[v, \lambda yx.\exists r((r \ \boldsymbol{q} \ C(\boldsymbol{x})) \ \&\ y = f\boldsymbol{x}r)] \ A(P))). \tag{8}$$

Fix $\boldsymbol{x}$ and r and assume

$$r \ \boldsymbol{q} \ C(\boldsymbol{x}).$$

We will show

$$f\boldsymbol{x}r \ \boldsymbol{q}_P[v, \lambda yx.\exists r((r \ \boldsymbol{q} \ C(\boldsymbol{x})) \ \&\ y = f\boldsymbol{x}r)] \ A(P). \tag{9}$$

By the assumption about q,

$$q\boldsymbol{x}r \ \boldsymbol{q} \ A(C).$$

Hence

$$q\boldsymbol{x}r \ \boldsymbol{q}_P[C, \lambda yx.(y \ \boldsymbol{q} \ C(\boldsymbol{x}))] \ A(P).$$

By positivity and $\forall \boldsymbol{x}(C(\boldsymbol{x}) \to v(\boldsymbol{x}))$,

$$q\boldsymbol{x}r \ \boldsymbol{q}_P[v, \lambda yx.(y \ \boldsymbol{q} \ C(\boldsymbol{x}))] \ A(P).$$

By letting $G := \lambda yx.(y\ \boldsymbol{q}\ C(x))$ in Lemma B.1,

$$\sigma^{P,f}_{A(P)}(qxr)\ \boldsymbol{q}_P[v, \lambda yx.\exists r((r\ \boldsymbol{q}\ C(x))\ \&\ y = fxr)]\ A(P).$$

By $fxr = \sigma^{P,f}_{A(P)}(qxr)$, we have

$$fxr\ \boldsymbol{q}_P[v, \lambda yx.\exists r((r\ \boldsymbol{q}\ C(x))\ \&\ y = fxr)]\ A(P).$$

Then (9) holds. Hence (8) holds. □

Appendix C: proof of stream extraction theorem

Lemma C.1. *Suppose that*

$$\forall x(\tilde{A}(x) \to N(nx)),$$

$$p \equiv \lambda x.nthcdr(nx_0)x,$$

$$A \equiv \nu X.\lambda x.x = \langle x_0, x_1\rangle\ \&\ \tilde{A}(x_0)\ \&\ X(px),$$

$$B \equiv \nu X.\lambda xy.\tilde{B}(x, y_0)\ \&\ X(px, y_1),$$

$$B^\circ \equiv \nu X.\lambda x.\exists z\tilde{B}(x, z)\ \&\ X(px).$$

Then

$$\forall f(\forall x(A(x) \to (fx\ \boldsymbol{q}\ B^\circ(x))) \to \forall x(A(x) \to B(x, filter(fx))))$$

holds.

Proof. By using only rules of NJ, the above goal is equivalent to

$$\forall xy\,(\exists f(\forall x(A(x) \to (fx\ \boldsymbol{q}\ B^\circ(x)))\ \&\ A(x)\ \&\ y = filter(fx)) \to B(x, y)).$$

By ($\nu 2$), it is sufficient to show

$$\forall xy(\exists f(\forall x(A(x) \to (fx\ \boldsymbol{q}\ B^\circ(x)))\ \&\ A(x)\ \&\ y = filter(fx)) \to$$
$$\tilde{B}(x, y_0)\ \&\ \exists g(\forall x(A(x) \to (gx\ \boldsymbol{q}\ B^\circ(x)))\ \&\ A(px)\ \&\ y_1 = filter(g(px)))).$$

By using only rules of NJ, it is equivalent to

$$\forall xf(\forall x(A(x) \to (fx\ \boldsymbol{q}\ B^\circ(x))\ \&\ A(x) \to$$
$$\tilde{B}(x, (filter(fx))_0)\ \&\ \exists g(\forall x(A(x) \to (gx\ \boldsymbol{q}\ B^\circ(x)))\ \&$$
$$A(px)\ \&\ (filter(fx))_1 = filter(g(px)))). \tag{10}$$

We will now prove this. Fix x and f and assume that

$$\forall x(A(x)) \to (fx\ \boldsymbol{q}\ B^\circ(x)), \tag{11}$$

$$A(x). \tag{12}$$

By (11) and (12), $(fx \; \boldsymbol{q} \; B^{\circ}(x))$ holds. Hence,

$$((fx)_{01} \; \boldsymbol{q} \; \tilde{B}(x,(fx)_{00})) \; \& \; ((fx)_1 \; \boldsymbol{q} \; B^{\circ}(px)) \tag{13}$$

holds. Therefore, $\tilde{B}(x,(filter(fx))_0)$ holds since $(filter(fx))_0 = (fx)_{00}$.

Put $g \equiv \lambda y.(f\langle x_0, \ldots, x_{nx_0-1}, y\rangle)_1$. We will show $\forall y(A(y) \to (gy \; \boldsymbol{q} \; B^{\circ}(y)))$. Fix y and assume that $A(y)$. By the definition of A and $p\langle x_0, \ldots, x_{nx_0-1}, y\rangle = y$,

$$A(x) \leftrightarrow x = \langle x_0, x_1\rangle \; \& \; \tilde{A}(x_0) \; \& \; A(px)$$

and

$$A(\langle x_0, \ldots, x_{nx_0-1}, y\rangle) \leftrightarrow \tilde{A}(x_0) \; \& \; A(y)$$

hold. By this and (12), $\tilde{A}(x_0)$ holds. Hence $A(\langle x_0, \ldots, x_{nx_0-1}, y\rangle)$ holds. Combining it with (11), we get $(f\langle x_0, \ldots, x_{nx_0-1}, y\rangle \; \boldsymbol{q} \; B^{\circ}(\langle x_0, \ldots, x_{nx_0-1}, y\rangle))$. Hence $((f\langle x_0, \ldots, x_{nx_0-1}, y\rangle)_1 \; \boldsymbol{q} \; B^{\circ}(y))$ and $(gy \; \boldsymbol{q} \; B^{\circ}(y))$ hold. Therefore we get $\forall y(A(y) \to (gy \; \boldsymbol{q} \; B^{\circ}(y)))$.

By (12), $A(px)$ holds. Since, in general, $(filter(s))_1 = filter(s_1)$ holds, we get $(filter(fx))_1 = filter((fx)_1) = filter(g(px))$. Therefore (10) holds. □

Proof of Theorem 6.3. By the assumptions and the definition of $\boldsymbol{q}$-realizability, $\forall x(A(x) \to (ex(jx) \; \boldsymbol{q} \; B^{\circ}(x)))$ holds. Letting f be $\lambda x.ex(jx)$ in Lemma C.1, we get $\forall x(A(x) \to B(x, Fx))$. □

References

[1] M. Beeson, *Foundations of Constructive Mathematics* (Springer, Berlin, 1985).

[2] R.L. Constable et al., *Implementing Mathematics with the Nuprl Proof Development System* (Prentice-Hall, Englewood Cliffs, NJ, 1986).

[3] P. Dybjer and H.P. Sander, A functional programming approach to the specification and verification of concurrent systems, *Formal Aspects Comput.* **1** (1989) 303–319.

[4] S. Hayashi and H. Nakano, PX: *A Computational Logic* (MIT Press, Cambridge, MA, 1988).

[5] S. Kobayashi, Inductive/coinductive definitions and their realizability interpretation, Manuscript, 1991.

[6] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[7] M. Tatsuta, Program synthesis using realizability, *Theoret. Comput. Sci.* **90** (1991) 309–353.

[8] M. Tatsuta, Monotone recursive definition of predicates and its realizability interpretation, *Proc. Theoretical Aspects of Computer Software*, Lecture Notes in Computer Science, Vol. 526 (Springer, Berlin, 1991) 38–52.